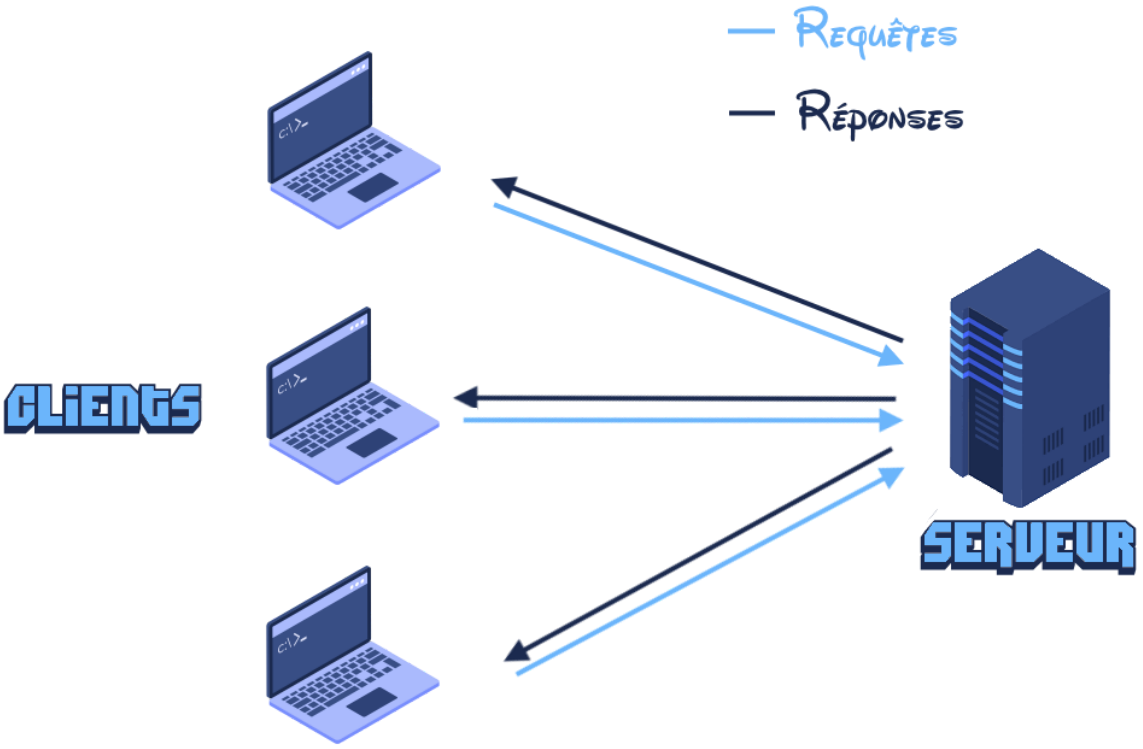
MONSEMPES Samuel

Projet Système & Réseaux
Décembre 2019



UGA
Université
Grenoble Alpes

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Sommaire

I/ Introduction

Le but de notre projet de systèmes et réseaux est de développer une application client/serveur avec le protocole TCP/IP. L'objectif de cette application est de permettre à un utilisateur de consulter la liste des trains disponibles selon certains critères (heure de départ, plage horaire) ainsi qu'une ville de départ et une ville d'arrivée.
L'application client devra interroger l'application serveur avec les critères de l'utilisateur. L'application serveur devra quant à elle répondre à la requête de tout les clients qui se connectent à elle.
Les différentes possibilités de requêtes attendues sont les suivantes :

- envoyer l'affichage du premier train possible à partir de l'horaire de départ demandée, ainsi que les villes de départ et d'arrivée.
- envoyer la liste des trains possibles dans une tranche horaire donnée par l'utilisateur, ainsi que les villes de départ et d'arrivée.
- envoyer la liste des trains avec seulement les villes de départ et d'arrivée

II/ Fonctionnalités réalisées

A) Manipulation du fichier

La manipulation du fichier de données se fait en parcourant le fichier texte ligne par ligne (une ligne correspond à un train), chaque ligne est parcourue et séparée en fonction du délimiteur ';'. Les données de chaque train sont stockées dans un tableau de chaînes de caractères puis analysées pour être traitées dans une *struct train.*

Le numéro d'identification est converti en *int*, le champ optionnel du fichier de données est converti en double (***REDUC*** = 0.8 ou ***SUPPL*** = 1.1 ou 1 si le champs est vide), ainsi que le prix. Notre variable *struct train* est ensuite ajoutée à un tableau de *struct train*.

B) Description de l'architecture serveur

À l'ouverture de l'application serveur, un socket d'écoute est créé et ensuite attaché. Une connexion est ouverte avec *listen*, le socket est alors en attente d'une demande de connexion. Il utilise alors la primitive *accept* pour accepter une demande de connexion. Dès lors qu'une demande de connexion est reçue et bien acceptée, l'application créée un processus fils qui prend le dialogue avec le client. Le processus père de l'application serveur repasse alors en attente d'une autre demande de connexion d'un autre client.

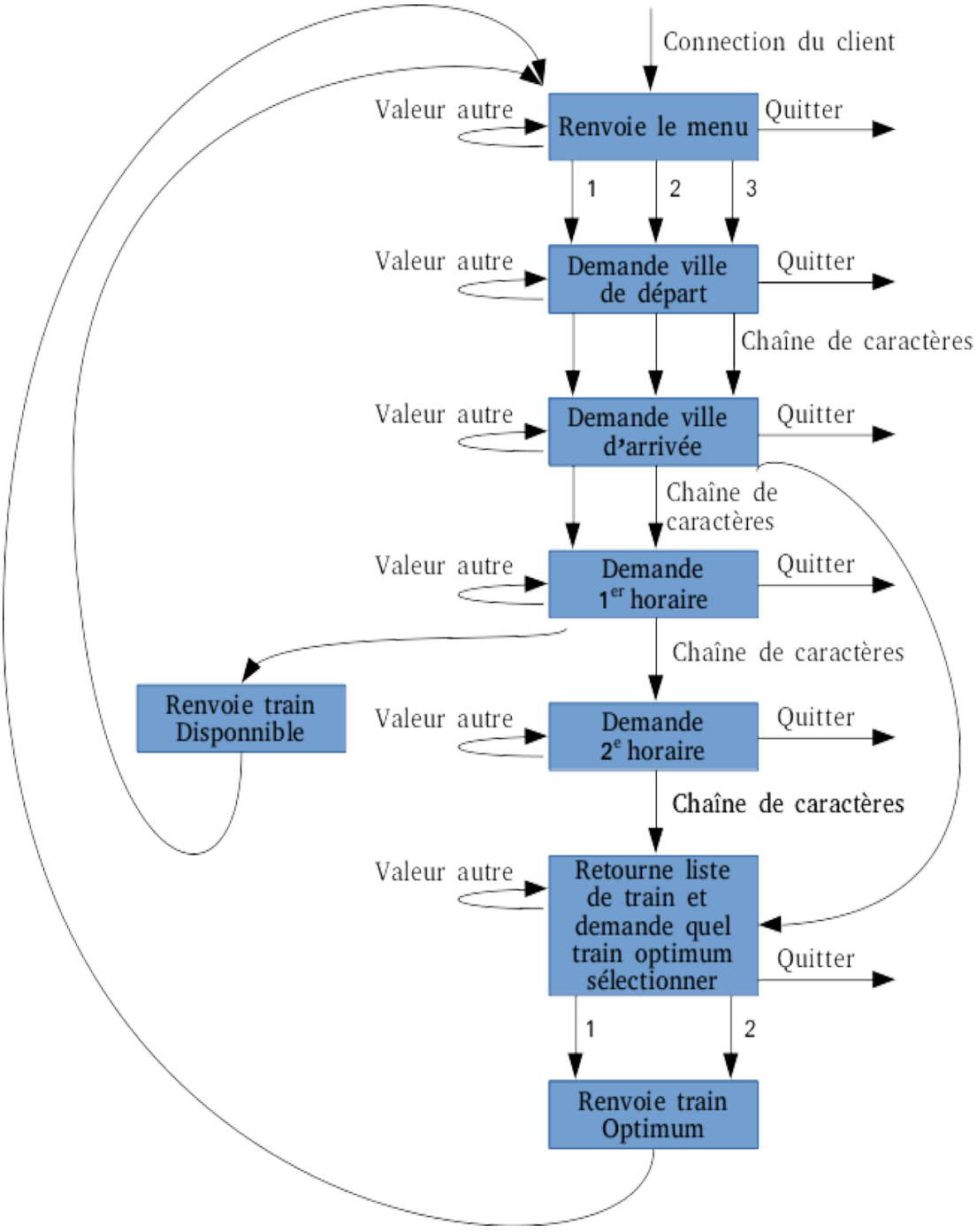
C) Description de l'architecture client

À l'ouverture de l'application client, un socket d'écoute est créé et demande à l'utilisateur le nom d'un serveur pour pouvoir récupérer son adresse avec *gethostbyname*. L'application fait alors une demande de connexion avec la primitive *connect*. Puis une fois la connexion établie, l'application client attend la réponse du serveur.

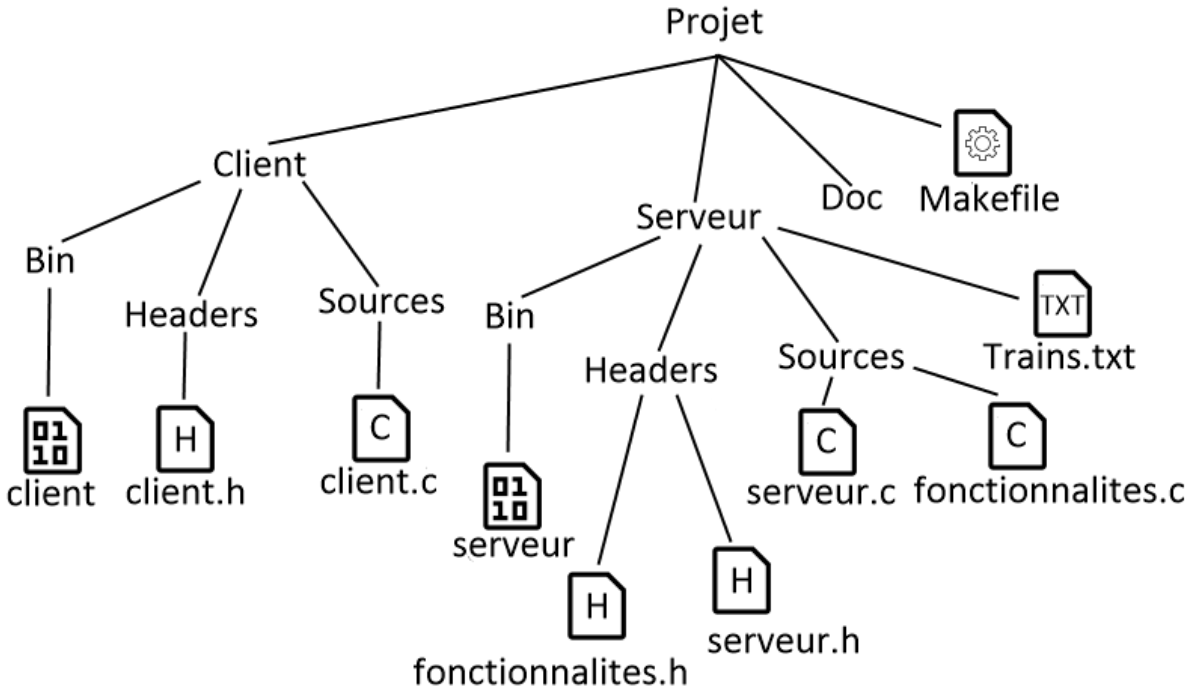
D) Fonctionnement du dialogue client/serveur

Le client se connecte au serveur, et se met en attente de réception d'un message de ce dernier. Le serveur reçoit la connexion et envoie le menu au client puis se met en attente de réception d'un message du client. Ainsi le dialogue fonctionne de la manière suivante: l'un parle puis attend la réponse, l'autre parle puis attend la réponse. Le client peut à tout moment stopper le dialogue en envoyant "Quitter", en faisant Ctrl+c ou tout simplement en fermant la fenêtre.

E) Fonctionnement du processus fils du serveur après connexion d'un client



F) Arborescence de fichiers



G) Fonctions principales du fichier fonctionnalites.c qui sont appelées par serveur.c

- Fonction1 : Cette fonction prend en paramètres une ville de départ, une ville d'arrivée et un horaire de départ. Elle retourne le premier train de la liste correspondant à ces trois critères.

- Fonction2 : Cette fonction prend en paramètres une ville de départ, une ville d'arrivée et une plage horaire de départ. Elle retourne un tableau contenant les trains disponibles satisfaisant les critères sélectionnés.

- Fonction3 : Cette fonction prend en paramètres une ville de départ et une ville d'arrivée. Elle retourne un tableau contenant les trains disponibles satisfaisant les critères sélectionnés.

- ChercherTempsOptimum : Cette fonction prend en paramètres un tableau de trains et la taille de ce tableau. Elle retourne le train ayant le trajet le plus court.

- ChercherCoutOptimum : Cette fonction prend en paramètres un tableau de trains et la taille de ce tableau. Elle retourne le train ayant le prix le plus bas.

- TrainToString : Cette fonction prend en paramètre un tableau de *struct train* et sa taille. Elle renvoie une chaîne de caractères contenant l'affichage des trains passés en paramètre. Elle est utilisée par chacune des fonctions ci-dessus pour pouvoir envoyer la chaîne de caractères contenant la réponse à la requête demandée par le client.

III/ Conception

A) Un client léger

Les avantages des clients légers sont multiples. J'ai sélectionné les principaux :

- **Déporter l'intelligence** car en effet, seul le serveur réalise les calculs complexes.
- **Faciliter les mises à jour** car uniquement le serveur doit être maintenu.
- **Homogénéité des versions accessibles** car tous les clients utilisent le même logiciel.
- **Réduction des coûts** (principalement matériels et sur le déploiement) car les clients coûtent peu cher et le nombre d'administrateurs est fortement diminué.
- Il est tout à fait possible de réaliser **exactement toutes les mêmes fonctionnalités** que peut faire un client lourd.
- **Un réseau beaucoup plus évolutif** : En effet, grâce à cette architecture il est possible de supprimer ou rajouter des clients beaucoup plus facilement et sans perturber le fonctionnement du réseau ni de faire de modification majeure.

C'est donc pour ces différentes raisons que j'ai choisis pour notre projet une conception basée sur un client dit "léger" plutôt qu'un client "lourd".

Source:

http://www-igm.univ-mlv.fr/~dr/XPOSE2007/pverron_client-leger/index.html?action=inventaire

https://fr.wikipedia.org/wiki/Client_léger
https://www.techwalla.com/articles/what-are-the-functions-of-client-server-computers-on-a-network

B) Un accès au fichier texte à chaque requête du client

Je suis parti du constat que le fichier texte était l'équivalent d'une base de données. Ainsi plutôt que de stocker à chaque nouvelle connexion la base de données dans un tableau de struct, j'ai préféré accéder au fichier à chacune des requêtes de trains.

IV/ Manuel d'utilisation

Avant toute utilisation, il est indispensable de se placer avec votre terminal à la racine du projet et exécuter le makefile. Ensuite, il faut exécuter les commandes suivantes dans deux terminaux distincts qui sont placés à la racine du projet :

"./serveur/bin/serveur"
"./client/bin/client"

Dans le terminal client, il vous faut tout d'abord renseigner le nom du serveur. Pour trouver le nom du serveur, il vous suffit de regarder sur le terminal du serveur dans le champ avant le $ ou utiliser l'IP du serveur.

Ensuite, l'utilisateur sélectionne la requête qu'il veut exécuter parmis les trois possibles présentées à l'aide d'un affichage de menu depuis le terminal côté client.
En fonction de son choix, des questions lui sont demandées pour pouvoir satisfaire tous les critères requis à l'exécution de la fonctionnalité demandée.

V/ Exemples d'expérimentation

- Les 5 fonctionnalités principales du projet sont mises en place ainsi que la fonctionnalité optionnelle de coloration.

```
Connecté au serveur.
Menu :
(1)       Recherche du premier train possible sur un trajet à partir d'un horaire
(2)       Recherche des trains possibles sur un trajet et une tranche horaire
(3)       Recherche des trains possibles sur un trajet
(Quitter)
Veuillez entrer 1, 2, 3 ou Quitter
1
Veuillez entrer la ville de départ :
Grenoble
Veuillez entrer la ville d'arrivée :
Valence
Veuillez entrer l'horaire de départ (de forme xx:xx ex : 13:50 pour treize heure cinquantes minutes) :
10:10
Train n°17524 de Grenoble à Valence partant à 16:30 et arrivant à 17:45 pour 17.60€
Menu :
(1)       Recherche du premier train possible sur un trajet à partir d'un horaire
(2)       Recherche des trains possibles sur un trajet et une tranche horaire
(3)       Recherche des trains possibles sur un trajet
(Quitter)
Veuillez entrer 1, 2, 3 ou Quitter
2
Veuillez entrer la ville de départ :
Valence
Veuillez entrer la ville d'arrivée :
Grenoble
Veuillez entrer l'heure à partir de laquelle vous souhaitez partir (de forme xx:xx ex : 13:50 pour treize heure cinquantes minutes) :
6:00
Veuillez entrer l'heure jusqu'à laquelle vous souhaitez partir (de forme xx:xx ex : 13:50 pour treize heure cinquantes minutes) :
12:00
Train n°17564 de Valence à Grenoble partant à 06:15 et arrivant à 07:31 pour 14.08€
Train n°17566 de Valence à Grenoble partant à 06:45 et arrivant à 07:55 pour 17.60€
Train n°17568 de Valence à Grenoble partant à 07:15 et arrivant à 08:32 pour 17.60€
Menu :
(1)       Recherche du  train le plus optimum niveau temps
(2)       Recherche du  train le plus optimum niveau prix
Veuillez entrer 1 ou 2
1
Train n°17566 de Valence à Grenoble partant à 06:45 et arrivant à 07:55 pour 17.60€
Menu :
(1)       Recherche du premier train possible sur un trajet à partir d'un horaire
(2)       Recherche des trains possibles sur un trajet et une tranche horaire
(3)       Recherche des trains possibles sur un trajet
(Quitter)
Veuillez entrer 1, 2, 3 ou Quitter
3
Veuillez entrer la ville de départ :
Valence
Veuillez entrer la ville d'arrivée :
Montelimar
Train n°86181 de Valence à Montelimar partant à 12:30 et arrivant à 12:56 pour 7.84€
Train n°86183 de Valence à Montelimar partant à 14:10 et arrivant à 14:30 pour 10.78€
Train n°86187 de Valence à Montelimar partant à 16:30 et arrivant à 16:56 pour 9.80€
Menu :
(1)       Recherche du  train le plus optimum niveau temps
(2)       Recherche du  train le plus optimum niveau prix
Veuillez entrer 1 ou 2
2
Train n°86181 de Valence à Montelimar partant à 12:30 et arrivant à 12:56 pour 7.84€
```

- Le processus fils se termine correctement dès que le client se ferme (Ctrl+c, "Quitter", ou fermer la fenêtre).
- Les différentes fonctions du programmes vérifient que les entrées du client correspondent aux attentes du serveur.

V/ Conclusion

A) Etat courant du programme:

L'application offre les possibilités suivantes :

- À la soumission d'une ville de départ, d'une ville d'arrivée, et d'un horaire de départ, obtient l'affichage du train satisfaisant ces critères, ou à défaut du premier train possible à partir de l'horaire de départ demandé.

- À la soumission d'une ville de départ, d'une ville d'arrivée, et d'une tranche horaire pour l'horaire de départ, obtient l'affichage des trains possibles dans cette tranche horaire.

- À la soumission d'une ville de départ et d'une ville d'arrivée, obtient l'affichage de la liste de tous les trains satisfaisant ces critères.

- L'application vous indique un train parmi cette liste de résultats, sélectionné selon l'un des critères suivants :
    - le trajet au meilleur prix.
    - le trajet de durée optimum.

- Les différentes fonctions du programme vérifient que les entrées du client correspondent aux attentes du serveur.

- Il y a une coloration syntaxique pour améliorer l'expérience utilisateur.

B) Pistes d'amélioration

- Améliorer le client afin qu'il interroge non pas un seul mais plusieurs serveurs de fournisseurs concurrents et sélectionne le meilleur résultat de tous ces fournisseurs.

- Proposer des fonctionnalités spéciales pour l'administration à distance : récolte de statistiques sur les consultations, mise à jour du fichier de données, ...